Advanced Prefix Sums

1D Prefix Sums:

Step 1: Create the prefix sum array

```
for (int i=1; i<=N; i++) {
        pre[i] = val[i] + pre[i-1];
}
```

Now pre[i] stores the sum of the first i elements.

Step 2: Answer the queries: O(1) per query

The sum of the range from a to b is:

```
pre[b] - pre[a-1]
```

1D Inverse Prefix Sums:

Step 1: Add all the ranges: O(1) per range

To add x to the range from a to b:

```
val[a] += x;
val[b+1] -= x;
```

Step 2: Make val[i] store the i'th value

```
for (int i=1; i<N; i++) {
        val[i] += val[i-1];
}
```

Now each value val[i] stores the correct amount at index i.

2D Prefix Sums:

Step 1: Create the prefix sum array

```
for (int i=1; i<=N; i++) {
	for (int j=1; j<=N; j++) {
		pre[i][j] = val[i][j] + pre[i-1][j] + pre[i][j-1] - pre[i-1][j-1];
	}
}
```

Step 2: Answer the queries: O(1) per query

The sum of the range from (x1, y1) to (x2, y2) is:

```
pre[x2][y2] - pre[x1-1][y2] - pre[x2][y1-1] + pre[x1-1][y1-1]
```

2D Inverse Prefix Sums:

Step 1: Add all the ranges

To add v to the range from (x1, y1) to (x2, y2):

```
val[x1][y1] += v;
val[x1][y2+1] -= v;
val[x2+1][y1] -= v;
val[x2+1][y2+1] += v;
```

Step 2: Make val[i][j] store the value at (i, j)

```
for (int i=1; i<=N; i++) {
	for (int j=1; j<=N; j++) {
		val[i][j] += + pre[i-1][j] + pre[i][j-1] - pre[i-1][j-1];
	}
}
```